MMVP: A Multimodal MoCap Dataset with Vision and Pressure Sensors

Supplementary Material

7. Dataset Details

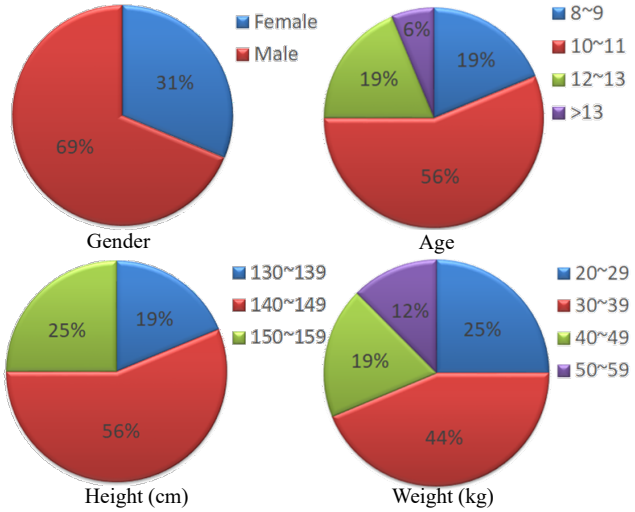

Figure 7. Statistics of MMVP subjects.

Synchronization: We uesed an Azure Kinect camera[1] and Xsensor pressure insoles (HX 210-510)[2] to capture the data. Due to this pressure insole's lack of automatic synchronization support, we have designed a coarse-to-fine synchronization method. The pressing switch controls the bulb. When the switch is pressed with the foot, the bulb illuminates, and the insole detects the pressure simultaneously, achieving coarse synchronization. Manual fine synchronization is performed by observing the sudden changes in pressure on the insole, such as landing after a long jump.

Statistics: The dataset consists of a total of 16 subjects, with 11 males and 5 females. Key statistics (gender, age, height, and weight) of the subjects are shown in Fig. 7. Synchronized RGBD frames, pressure data, and registrations are more than 44k frames. The dataset consists of high-speed and large-range movements, including skipping rope, long jump, ball throwing, side stepping, running, and dancing.

Ethics: The subjects in MMVP are well-informed and have voluntarily signed legal agreements to allow the data to be made public for research purposes.

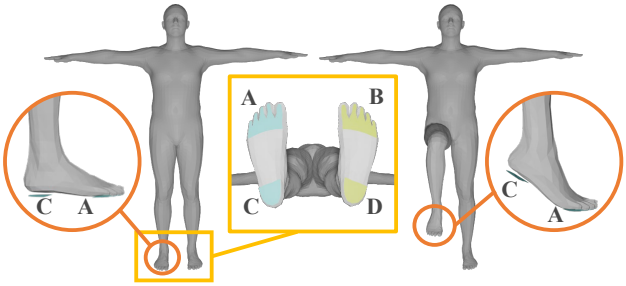

Figure 8. foot plane details.

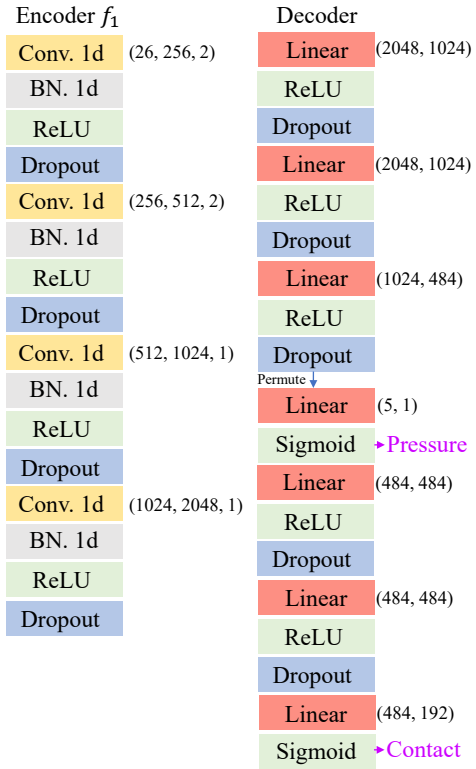

Figure 9. FPP-Net details.

8. RGBD-P fitting

8.1. Foot Contact Temporal Loss

We design four-foot planes derived from SMPL template mesh, as shown in Fig. 8. By projecting the foot vertices of the template SMPL model onto the ground, we acquire

[1]https://azure.microsoft.com/en-us/products/kinect-dk/

[2]https://www.xsensor.com/solutions-and-platform/human-performance/gait-motion-insoles

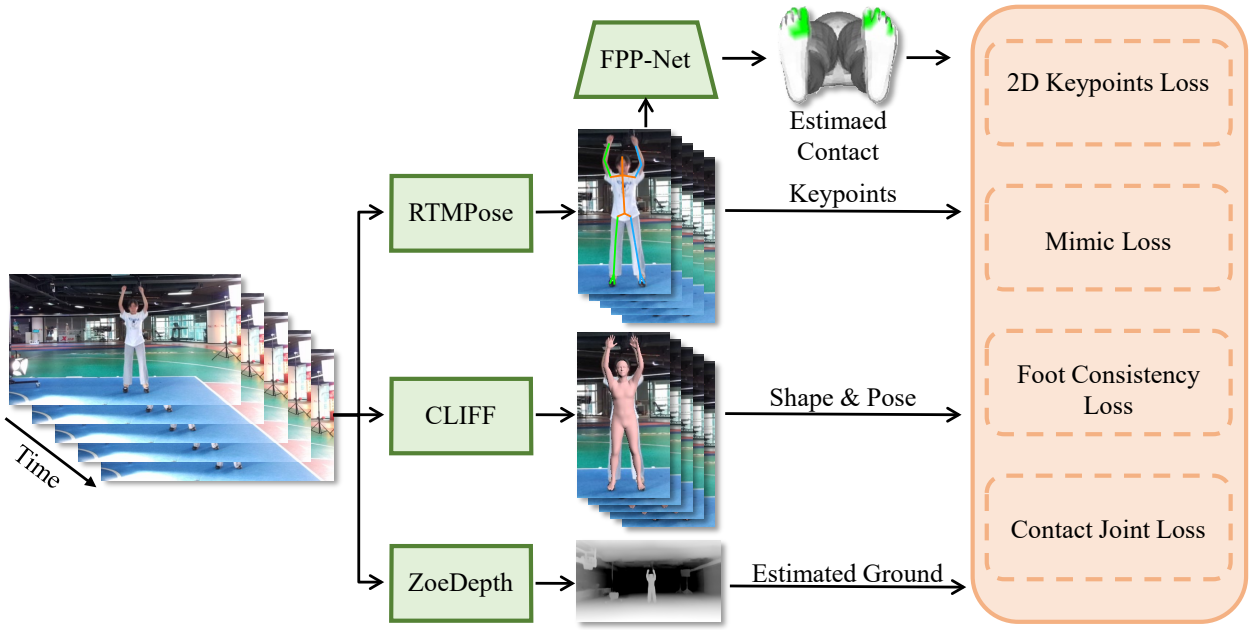

Figure 10. Flowchart of pose and translation optimization.

the anterior and posterior foot planes. The points on foot planes, which are associated with SMPL foot vertices, can also be controlled by SMPL parameters.

When computing temporal contact constraints, we first convert the contact of SMPL vertices into contact of points on the plane. Then, we select the points in each plane whose spatial position should remain static between adjacent frames. Subsequently, we calculate the L2 loss of the position of these selected points between frame $t-1$ and frame t. By minimizing this loss, we ensure that the selected points on the foot surface maintain consistent spatial positions over time.

When describing the state of the foot, foot planes can be more effective compared to foot surface vertices. These planes can provide valuable information about the orientation and inclination of the foot relative to the ground, which can aid in accurately controlling the foot pose. By utilizing this information from foot planes, we can enhance the precision and accuracy of foot pose control.

8.2. RGBD-P Fitting Implementation Details

The optimization steps for ground truth generating can be divided into three parts: shape fitting, pose initialization, and pose tracking.

The first step is to fit the shape of the human body. We optimize the shape-related parameters (α, β) for each character. Specifically, we use the corresponding A-pose sequence from the dataset to perform this optimization.

The second step is to fit the initial pose parameter for each action sequence. For each action sequence, we select a specific frame, denoted as t_0, as the initial moment for pose tracking. Utilizing the shape-related parameters (α, β) calculated in the first step, we optimize the pose parameters $(\theta_0, \boldsymbol{T}_0)$ for frame t_0.

The third step is to generate pose ground truth of each motion sequence. We leverage the shape-related parameters (α, β) and the initial pose parameters $(\theta_0, \boldsymbol{T}_0)$ to estimate the pose for the entire motion sequence. During our tracking process at frame t, $(\theta_{t-1}, \boldsymbol{T}_{t-1})$ will be applied in fitting the pose parameters $(\theta_t, \boldsymbol{T}_t)$.

The relevant methods are implemented using Python. We utilize Adam to optimize Eq. (12) within PyTorch.

8.3. RGBD-P fitting Comparison Details

We compare our results with PROX [3] and LEMO [4] on MMVP Dataset. All configs are set as their default values. We use four metrics to evaluate fitting performance: E_{3d}, Mean Foot-Contact Error (MFCE), F1 score, and IOU. For the last three metrics, given a fitting result, we set a fixed threshold to calculate the foot contact $C_{fitting}$ as POSA [15]. We calculate the F1 score and IOU between $C_{fitting}$ and the ground truth contact C. The Mean Foot-Contact Error (MFCE) is calculated as $\|C_{fitting} - C\|_2$.

Compared to PROX, our method considers body shape differences between adults and children, ensuring shape parameter consistency in one motion sequence. PROX sets a distance threshold between foot and ground, which may result in sudden motion. To avoid this, our method leverages

[3] https://github.com/mohamedhassanmus/prox
[4] https://github.com/sanweiliti/LEMO

dense foot-ground distance as loss function in optimization. Based on the fitting results of PROX, LEMO considers temporal motion changing and contact consistency, however, it neglects the alignment with depth data, which can lead to difficulties in accurately estimating global position.

9. VP-MoCap

9.1. FPP-Net

Network details. We follow PhysCap [5] and Jesse *et al*. [44] to construct our network. The details are shown in Fig. 9. The numbers next to "Conv.1d" represent the input dimension, output dimension, and kernel size. The numbers next to "Linear" represent the input dimension and output dimension. The outputs of FPP-Net are marked in purple. We train our network for 20 hours (900 epochs). The initial learning rate is 10^{-4}. The optimizer is Adam.

Dataset split. We split 16 subjects in MMVP into 13 and 3 for training and testing, respectively.

BSTRO fintune details. We fintune BSTRO[6] with MMVP-test on their pre-trained model. All configurations are set to default values.

9.2. Pose and Translation Optimization Details

As shown in Fig. 10, our pose and translation optimization is constructed using several inputs, including the estimated contact, human body keypoints, initial shape and pose, and estimated ground. The estimated contact is estimated through FPP-Net, utilizing human body keypoints as input, which are estimated by RTMPose [20]. The initial shape and pose are estimated by the off-the-shelf method CLIFF [31]. Additionally, the estimated ground is obtained through the application of ZoeDepth [2]. We used VPoser [39] as human body prior.

9.3. Pose and Translation Optimization Comparison Details

We unify the focal length across our method and other comparative methods, maintaining the remaining settings of CLIFF [7], TRACE [8] and SMPLer-X [9] consistent with default parameters.

9.4. More Results on In-the-Wild Datasets

We compared the results on in-the-wild datasets, 3DPW [53] and EMDB [24]. Our method is mainly designed for videos from static viewpoints, so it remains challenging to handle moving camera configurations. Therefore, we selected sequences with relatively static camera movements in 3DPW and EMDB (1K frames from 3DPW and 2.5K frames from EMDB) for conducting the comparison. Note that 3DPW does not provide GT global trajectory annotations for evaluation. As shown in Tab. 5 and Fig. 11, our method achieved significant improvement in global translation estimation on EMDB (3DPW does not provide GT trans) and also produces slightly better pose estimation results.

	Methods	M. ↓	PM. ↓	PVE ↓	PF. ↓	Traj ↓
3DPW	CLIFF	85.2	24.6	108.6	153.2	-
	TRACE	99.4	37.2	127.0	185.0	-
	SMPLer-X	109.3	30.1	137.2	197.1	-
	Ours	81.6	41.3	106.4	129.4	-
EMDB	CLIFF	93.8	39.7	129.0	146.7	105.2
	TRACE	103.0	41.6	133.4	197.8	114.5
	SMPLer-X	95.8	35.6	127.0	160.7	4943.0
	Ours	89.0	37.3	125.9	115.1	58.8

Table 5. Evaluation of pose and translation estimation on 3DPW and EMDB.M.: MPJPE, PM. : PMPJPE, PF. : PVE (Feet).

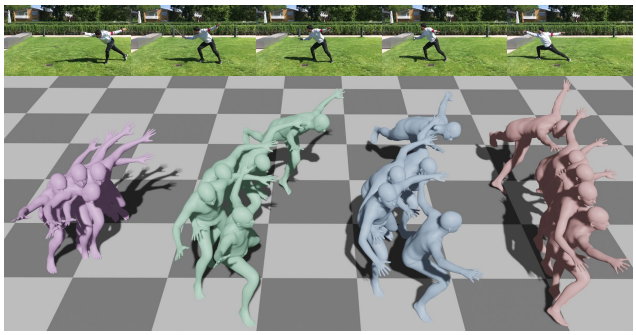

Figure 11. Comparison of 3D translation estimation in 3DPW. From left to right: SMPLer-X (Purple), TRACE (Green), CLIFF (Blue), and our method (Pink).

[5] https://github.com/soshishimada/PhysCap_demo_release
[6] https://github.com/paulchhuang/bstro
[7] https://github.com/huawei-noah/noah-research/tree/master/CLIFF
[8] https://github.com/Arthur151/ROMP/tree/master/simple_romp/trace2
[9] https://github.com/caizhongang/SMPLer-X